

Çanakkale Onsekiz Mart Üniversitesi, Mühendislik Fakültesi,
Bilgisayar Mühendisliği Akademik Dönem 2022-2023
Ders: BLM-4014 Yapay Sinir Ağları/Bahar Dönemi
ARA SINAV SORU VE CEVAP KAĞIDI
Dersi Veren Öğretim Elemanı: Dr. Öğretim Üyesi Ulya Bayram

Öğrenci Adı Soyadı: Ömer Faruk Akyapak
Öğrenci No: 190401008

14 Nisan 2023

Açıklamalar:

- Vizeyi çözüp, üzerinde aynı sorular, sizin cevaplar ve sonuçlar olan versiyonunu bu formatta PDF olarak, Teams üzerinden açtığım assignment kısmına yüklemeniz gerekiyor. Bu bahsi geçen PDF'i oluşturmak için LaTeX kullandıysanız, tex dosyasının da yer aldığı Github linkini de ödevin en başına (aşağı url olarak) eklerseniz bonus 5 Puan! (Tavsiye: Overleaf)
- Çözümlerde ya da çözümlerin kontrolünü yapmada internetten faydalanmak, ChatGPT gibi servisleri kullanmak serbest. Fakat, herkesin çözümü kendi emeğinden oluşmak zorunda. Çözümlerinizi, cevaplarınızı aşağıda belirttiğim tarih ve saate kadar kimseyle paylaşmayınız.
- Kopyayı önlemek için Github repository'lerinizin hiçbirini **14 Nisan 2023, saat 15:00'a kadar halka açık (public) yapmayınız!** (Assignment son yükleme saati 13:00 ama internet bağlantısı sorunları olabilir diye en fazla ekstra 2 saat daha vaktiniz var. **Fakat 13:00 - 15:00 arası yüklemelerden -5 puan!**)
- Ek puan almak için sağlayacağımız tüm Github repository'lerini **en geç 15 Nisan 2023 15:00'da halka açık (public) yapmış olun linklerden puan alabilmek için!**
- **14 Nisan 2023, saat 15:00'dan sonra gönderilen vizeler değerlendirilmeye alınmayacak, vize notu olarak 0 (sıfır) verilecektir!** Son anda internet bağlantısı gibi sebeplerden sıfır almayı önlemek için assignment kısmından ara ara çözümlerinizi yükleyebilirsiniz yedekleme için. Verilen son tarih/saatte (14 Nisan 2023, saat 15:00) sistemdeki en son yüklü PDF geçerli olacak.
- Çözümlerin ve kodların size ait ve özgün olup olmadığını kontrol eden bir algoritma kullanılacaktır. Kopya çektiği belirlenen vizeler otomatikman 0 (sıfır) alacaktır. Bu nedenle çözümlerinizi ve kodlarınızı yukarıda sağladığım gün ve saatlere kadar kimseyle paylaşmayınız.
- Bu vizeden alınabilecek en yüksek not 100'dür. Toplam aldığımız puan 100'ü geçerse, aldığımız not 100'e sabitlenecektir.
- LaTeX kullanarak PDF oluşturanlar öz geçmişlerine LaTeX bildiklerini de eklemeyi unutmasınlar :)
- Bu vizedeki soruların çözümleri ve tex dosyası için istediğiniz kadar sayıda Github repository'si oluşturabilirsiniz. Sadece yukarıda belirttiğim tarihlerde (14 Nisan 2023 saat 15:00 ile 15 Nisan 2023 saat 15:00 arasında) public yapmayı/halka açmayı ve Github profilinizi de öz geçmişinize eklemeyi unutmayın :)
- Bilgisayarınıza hiçbir program kurmadan, Overleaf, Google Colab, Kaggle gibi ortamlar üzerinden bu vizeyi tamamlamanız mümkün. İyi çalışmalar!!

(Ekstra 5 puan) <https://github.com/OmerFarukAkyapak/yapaysiniraglari>

1 (Toplam 10 Puan) Hopfield Networks:

1.1 (2 Puan) ChatGPT'ye size Hopfield Network'ler ile ilgili detaylı bilgi vermesini isteyin. Sorduğunuz soruyu, ChatGPT'nin çıkarttığı cevabı aşağı koyun. Bu cevapta anlamadığınız terimleri de ChatGPT'ye sorup, sorduğunuz soruları ve ChatGPT'nin verdiği cevapları da sırayla aşağı koyun.

Hopfield ağı, asenkron bir geri beslemeli sinir ağıdır ve özellikle hafıza ve optimizasyon problemlerini çözmek için kullanılır. İlk olarak John Hopfield tarafından 1982'de tanıtılmıştır.

Hopfield ağları, bir dizi bağlantılı düğümden (nöron) oluşan bir ağ yapısına sahiptir. Her düğüm, diğer düğümlere bağlanan ağırlıklı bağlantılarla ilişkilendirilmiştir. Her düğüm, girdi sinyallerini alır, bunları işler ve bir çıkış sinyali üretir.

Hopfield ağları, iki temel işlevi gerçekleştirir: hafıza ve asenkron dinamik. Hafıza özelliği sayesinde, ağ belirli desenleri hatırlayabilir ve daha sonra bu desenleri hatırladığı girdilerle eşleştirebilir. Asenkron dinamik özelliği, ağın asenkron bir şekilde güncellenmesine ve her düğümün zamanla dinamik olarak değişmesine olanak tanır.

Hopfield ağları, bir enerji fonksiyonu üzerinden çalışır. Enerji fonksiyonu, ağın belirli bir durumunun enerjisini temsil eder ve ağ, enerjiyi en aza indirmeye çalışır. Bu, ağın, enerji fonksiyonunun lokal minimum noktalarına doğru ilerlerken bir iterasyon süreci ile güncellendiği dinamik davranışını açıklar.

Hopfield ağları, özellikle hafıza ve optimize etme problemleri için kullanılır, örneğin desen tanıma, veri sıkıştırma ve kombinatoriyal optimizasyon gibi uygulamalarda başarıyla kullanılabilirler.

Terimler:

Kombinatoriyal optimizasyon: Birçok olası çözüm kombinasyonu arasından en iyi çözümü bulmak için matematiksel ve hesaplamalı yöntemlerin kullanıldığı bir optimizasyon alt dalıdır. Kombinatoriyal optimizasyon problemleri, belirli bir problemi en iyi şekilde çözmek için bir dizi karar değişkeni veya parametre arasından seçim yapma ihtiyacını içerir.

Asenkron geri beslemeli sinir ağı: Sinir ağlarının bir türüdür ve geri besleme bağlantıları içerir. Geri beslemeli sinir ağları, çıkışların daha önceki durumlarına bağlı olarak güncellendiği ve girdi sinyallerini tekrar kullanarak iteratif bir şekilde güncellendiği ağlardır.

1.2 (8 Puan) ChatGPT'nin açıklamalarından ne anladığınızı buraya kendi kelimelerinizle özetleyin. Örneğin "Hopfield Network nedir? Ne işe yarar? Neden bilmemiz gerekir bunu? Gerçek hayatta kullanılıyor mu?" gibi. Anlamadığınız kısımlar varsa ve ChatGPT'nin açıklamaları yeterli gelmezse internet dahil farklı kaynaklardan araştırıp, bilgilerin doğruluğunu sorgulamakta serbestsiniz. Konuyu doğru anlayıp anlamamanız puana etki edecektir. Kullandığınız kaynakları da belirtin!

Nedir?

Asenkron bir geri beslemeli sinir ağıdır. Yani bir tür sinir ağıdır ve bu türün özelliği yapılan çıkışlar sonucunda oluşan değerleri tekrar tekrar kullanıp artımlı bir şekilde durumlar güncellenir.

Ne işe yarar?

Hopfield ağları, genel olarak hafıza, desen tanıma, veri sıkıştırma, kombinatoriyal optimizasyon ve en yakın komşu eşleştirmesi gibi problemlerde kullanılabilir ve birçok farklı uygulama alanında potansiyel çözümler sunabilir.

Neden bilmeliyiz?

Hopfield ağları, örüntü tanıma ve depolama yoluyla insan hafızasını simüle etme kavramı ile ilişkilidir. Bununla birlikte birçok soruna çözüm sunar. Hafıza, optimizasyon problemlerine çözüm bulabilmek için bilmeliyiz.

Gerçek hayatta kullanılıyor mu?

Şifreleme işlemlerinde, karakter-desen eşleştirmede, bozuk şekillerin yeniden oluşturulmasında, el yazı tanımlama sistemlerinde kullanılmaktadır.

Kaynak:

ChatGPT

<https://medium.com/@batincangurbuz/hopfield-ağ-modeli-hopfield-network-hn-ccf1548ca432>

<https://tr.theastrologypage.com/hopfield-network>

<https://link.springer.com/book/9780387310732>

2 (Toplam 10 Puan) Boltzman Machine:

2.1 (2 Puan) ChatGPT’ye size Boltzman Machine ilgili detaylı bilgi vermesini isteyin. Sorduğunuz soruyu, ChatGPT’nin çıkarttığı cevabı aşağı koyun. Bu cevapta anlamadığınız terimleri de ChatGPT’ye sorup, sorduğunuz soruları ve ChatGPT’nin verdiği cevapları da sırayla aşağı koyun.

Boltzmann Machine (Boltzmann Makinesi), yapay sinir ağları alanında bir tür olası durum ağı (probabilistic graphical model) olan enerji tabanlı bir istatistiksel modeldir. Boltzmann Makinesi, ağı gizli ve görünür (gözlemlenen) katmanlarını içeren iki katmanlı bir ağ yapısına sahiptir ve bu katmanlar arasındaki bağlantılar çift yönlüdür. Boltzmann Makinesi, her bir gizli ve görünür birimi belirli bir olasılıkla etkinleştirilebilir veya etkisizleştirilebilir. Bu nedenle, Boltzmann Makinesi, olası durumların geniş bir yelpazesini temsil edebilir ve genel olarak olasılık temelli bir modele dayanır.

Boltzmann Makinesi, ağıdaki gizli ve görünür birimlerin durumlarını, enerji fonksiyonu olarak adlandırılan bir fonksiyonla temsil eder. Enerji fonksiyonu, ağı parametreleri tarafından belirlenir ve ağı belirli bir durumdaki enerji seviyesini hesaplar. Boltzmann Makinesi, enerji fonksiyonu kullanarak ağı durumları arasında geçişleri olasılıklarla modelleyen bir stokastik süreç olarak çalışır. Belirli bir durumdaki ağı olasılığını hesaplamak için Boltzmann dağılımı kullanılır.

Boltzmann Makinesi, Markov zinciri Monte Carlo (MCMC) yöntemleri gibi özel algoritmalara dayanarak ağı istatistiksel özelliklerini öğrenme yeteneğine sahiptir. Öğrenme süreci, ağı enerji fonksiyonunu optimize etmek için veriye dayalı bir süreçtir. Öğrenme tamamlandığında, Boltzmann Makinesi, ağı gizli birimlerini kullanarak verileri yeniden üretebilir ve bu nedenle veri üretimi, örnekleme, hatta modelleme ve tahminleme gibi bir dizi uygulamada kullanılabilir. Boltzmann Makinesi, derin öğrenme alanının temel taşlarından biridir ve birçok farklı uygulama alanında kullanılır, örneğin görüntü işleme, doğal dil işleme, öneri sistemleri ve karmaşık veri modellenmesi gibi alanlarda başarıyla uygulanmıştır.

Terimler:

Markov zinciri Monte Carlo (MCMC): Karmaşık olasılık dağılımlarını keşfetmek için kullanılan bir istatistiksel simülasyon yöntemidir. MCMC, istatistiksel çıkarımlar ve model tahminlemesi gibi birçok uygulama alanında kullanılır.

Regresyon: Bir değişkenin diğer bir veya daha fazla değişkenle ilişkisini inceleyen bir istatistiksel analiz yöntemidir. Genellikle bir bağımlı değişkenin, bir veya daha fazla bağımsız değişkenle ilişkisini modellemek ve tahmin etmek için kullanılır.

2.2 (8 Puan) ChatGPT’nin açıklamalarından ne anladığınızı buraya kendi kelimelerinizle özetleyin. Örneğin “Boltzman Machine nedir? Ne işe yarar? Neden bilmemiz gerekir bunu? Gerçek hayatta kullanılıyor mu?” gibi. Anlamadığınız kısımlar varsa ve ChatGPT’nin açıklamaları yeterli gelmezse internet dahil farklı kaynaklardan araştırıp, bilgilerin doğruluğunu sorgulamakta serbestsiniz. Konuyu doğru anlayıp anlamamanız puana etki edecektir. Kullandığınız kaynakları da belirtin!

Nedir?

Sınıflandırma, regresyon ve özellik öğrenimi işlemlerini yapan Boltzmann makinesi giriş veri seti üzerinde olasılıksal dağılımları öğrenebilen bir sinir ağıdır.

Ne işe yarar?

Zor kombinatoriyal problemleri temsil edip çözebilirler. Boltzmann makineleri, makine öğrenmesi veya çıkarımında pratik sorunlar için yararlı değildir, ancak eğer bağlantı düzgün bir şekilde kısıtlıysa, öğrenme pratik problemler için faydalı olacak kadar verimli hale getirilebilir.

Neden bilmeliyiz?

Boltzmann makinesi hakkında bilgi sahibi olmak, makine öğrenimi alanında daha derin bir anlayış geliştirmemizi, enerji tabanlı modelleme ve stokastik hesaplamalar hakkında daha fazla bilgi edinmemizi ve çeşitli uygulama alanlarında model seçimi ve kullanımında avantaj elde etmemizi sağlayabilir.

Gerçek hayatta kullanılıyor mu?

Boltzmann makinesi, genellikle teorik bir model olarak kullanılmaktadır ve gerçek hayatta doğrudan uygulanması nadir görülmektedir. Örneğin, Gibbs Örnekleme adlı bir yöntem, Boltzmann makinesinden türetilmiş bir örnekleme yöntemidir ve Markov Zinciri Monte Carlo (MCMC) adlı bir istatistiksel yöntemde sıkça kullanılır. Ancak, Boltzmann makinesinin doğrudan pratik uygulamalarda kullanılması yaygın değildir. Çoğunlukla teorik ve araştırma düzeyinde kullanılmaktadır ve gerçek hayatta kullanımı sınırlıdır.

Kaynak:

ChatGPT

<https://medium.com/@batincangurbuz/boltzman-makinesi>

<https://tr.theastrologypage.com/boltzmann-machine>

<https://devhuntery.wordpress.com/2018/07/25/kisitli-boltzmann-makineleri/>

3 (Toplam 10 Puan) Markov Assumption ve Markov Chain:

3.1 (2 Puan) ChatGPT'ye size Markov assumption ve Markov Chain ilgili detaylı bilgi vermesini isteyin. Sorduğunuz soruyu, ChatGPT'nin çıkarttığı cevabı aşağı koyun. Bu cevapta anlamadığınız terimleri de ChatGPT'ye sorup, sorduğunuz soruları ve ChatGPT'nin verdiği cevapları da sırayla aşağı koyun.

Markov ön kabulü (Markov assumption), bir sürecin gelecekteki durumunun sadece şu anki duruma bağlı olduğunu ve geçmiş durumlardan bağımsız olduğunu varsayan bir istatistiksel ön kabuldür. Bu ön kabul, Markov zinciri adı verilen bir süreç sınıfını temsil eder.

Markov zinciri (Markov Chain), belirli bir durum kümesinde zaman içinde geçişlerin rastgele olarak gerçekleştiği stokastik bir süreçtir. Markov zinciri, bugünkü durumun, yalnızca bir önceki duruma bağlı olduğu bir süreçtir ve geçmiş durumlardan bağımsızdır. Bu nedenle, Markov zinciri, gelecekteki durumları tahminlemek için sadece şu anki durumu kullanır.

Markov zincirleri, birçok uygulama alanında kullanılır, örneğin zaman serileri analizi, doğal dil işleme, makineler arası iletişim ve rastgele süreçlerin modellenmesi gibi alanlarda yaygın olarak kullanılır. Markov zincirleri, olasılık teorisi ve istatistik alanlarında önemli bir konudur ve birçok farklı matematiksel ve istatistiksel yöntemle incelenmiştir.

Terimler:

Markov zincirleri: Zaman içinde belirli bir durum kümesi üzerinde rastgele geçişlerin gerçekleştiği stokastik bir süreçtir. Örneğin, zaman serileri analizi, doğal dil işleme, makineler arası iletişim ve rastgele süreçlerin modellenmesi gibi alanlarda Markov zincirleri yaygın olarak kullanılır.

3.2 (8 Puan) ChatGPT’nin açıklamalarından ne anladığınızı buraya kendi kelimelerinizle özetleyin. Örneğin “Markov assumption ve Markov Chain nedir? Ne işe yarar? Neden bilmemiz gerekir bunu? Gerçek hayatta kullanılıyor mu?” gibi. Anlamadığınız kısımlar varsa ve ChatGPT’nin açıklamaları yeterli gelmezse internet dahil farklı kaynaklardan araştırıp, bilgilerin doğruluğunu sorgulamakta serbestsiniz. Konuyu doğru anlayıp anlamamanız puana etki edecektir. Kullandığınız kaynakları da belirtin!

Nedir?

Markov varsayımı, bir süreç veya sistemin mevcut durumunun, sadece anlık durumundan ve birkaç önceki durumundan bağımsız olduğunu ifade eden bir istatistiksel varsayımdır.

Ne işe yarar?

Markov varsayımı ve Markov zincirleri, birçok uygulama alanında kullanılan önemli istatistiksel araçlardır ve çeşitli problemleri modellemek ve çözmek için kullanılabilirler.

Neden bilmeliyiz?

Zaman serileri analizi, Doğal dil işleme, Hesaplamalı istatistik, Rastgele süreçlerin modellenmesi, Makine öğrenimi gibi yapay zekanın kullanıldığı alanlarda durumların modellenmesi için bilmemizde fayda vardır.

Gerçek hayatta kullanılıyor mu?

Kullanılıyor. Google’ın kullandığı internet sayfalarının PageRank’i Markov zinciri ile tanımlanmaktadır. Markov modelleri aynı zamanda kullanıcıların internet gezinti davranışlarını analiz etmek için de kullanılmıştır. Bir kullanıcının bir internet sayfasından web bağlantısı ile geçişi, birincil ya da ikincil Markov modelleri ile modellenabilir ve gelecekteki hareketleri öngörmede ve dolayısıyla internet sayfasını kullanıcı için kişiselleştirme de kullanılabilir. Metin üretiminden finansal modellemeye kadar birçok farklı alanda kullanılmıştır. Fakat en çok kullanılan alanları ise metin oluşturma ve otomatik tamamlama uygulamalarıdır.

Kaynak:

ChatGPT

<https://medium.com/@batincangurbuz/markov-zinciri-markov-chain-mc-33cd8a61f6fa>

<https://egealpay1.medium.com/hidden-markov-models-sakli-markov-modeli-b381380d0aca>

<https://bilgisayarkavramlari.com/2009/06/17/markof-modeli-markov-model/>

<https://tr.wikipedia.org/wiki/Markovzinciri>

4 (Toplam 20 Puan) Feed Forward:

- Forward propagation için, input olarak şu X matrisini verin (tensöre çevirmeyi unutmayın):

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \text{ Satırlar veriler (sample'lar), kolonlar öznitelikler (feature'lar).}$$

- Bir adet hidden layer olsun ve içinde tanh aktivasyon fonksiyonu olsun
- Hidden layer’da 50 nöron olsun
- Bir adet output layer olsun, tek nöronu olsun ve içinde sigmoid aktivasyon fonksiyonu olsun

Tanh fonksiyonu:

$$f(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Sigmoid fonksiyonu:

$$f(x) = \frac{1}{1+\exp(-x)}$$

Pytorch kütüphanesi ile, ama kütüphanenin hazır aktivasyon fonksiyonlarını kullanmadan, formülünü verdiğim iki aktivasyon fonksiyonunun kodunu ikinci haftada yaptığımız gibi kendiniz yazarak bu yapay sinir ağını oluşturun ve aşağıdaki üç soruya cevap verin.

4.1 (10 Puan) Yukarıdaki yapay sinir ağını çalıştırmadan önce pytorch için Seed değerini 1 olarak set edin, kodu aşağıdaki kod bloğuna ve altına da sonucu yapıştırın:

```
class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()
        self.hidden = nn.Linear(3, 50) # Creates a linear layer with input size 3
                                         and output size 50
        self.output = nn.Linear(50, 1) # Creates a linear layer with input size 50
                                         and output size 1
        self.tanh = nn.Tanh() # Creates a Tanh activation function layer that
                               applies Tanh activation function to inputs
        self.sigmoid = nn.Sigmoid() # Creates a Sigmoid activation function layer
                                     that applies Sigmoid activation
                                     function to inputs

    def tanh(self, x):
        return (torch.exp(x) - torch.exp(-x)) / (torch.exp(x) + torch.exp(-x)) #
                                             Computes the Tanh activation
                                             function

    def sigmoid(self, x):
        return 1 / (1 + torch.exp(-x)) # Computes the Sigmoid activation function

    def forward(self, x):
        x = self.tanh(self.hidden(x)) # Applies Tanh activation function to inputs
                                       and applies the hidden layer
        x = self.sigmoid(self.output(x)) # Applies Sigmoid activation function to
                                         inputs and applies the output
                                         layer

        return x # Returns the output

torch.manual_seed(1) # Sets the random seed of the torch module to 1 for
                     reproducibility
net = Net() # Creates an instance of the 'Net' class called 'net'
output = net(X) # Computes an output ('output') using the 'net' object with input 'X'
print(output)
```

Output:

tensor([[0.4892], [0.5566]])

4.2 (5 Puan) Yukarıdaki yapay sinir ağını çalıştırmadan önce Seed değerini öğrenci numaranız olarak değiştirip, kodu aşağıdaki kod bloğuna ve altına da sonucu yapıştırın:

```
class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()
        self.hidden = nn.Linear(3, 50) # Creates a linear layer with input size 3
                                         and output size 50
```

```

        self.output = nn.Linear(50, 1) # Creates a linear layer with input size 50
                                         and output size 1
        self.tanh = nn.Tanh() # Creates a Tanh activation function layer that
                               applies Tanh activation function
                               to inputs
        self.sigmoid = nn.Sigmoid() # Creates a Sigmoid activation function layer
                                      that applies Sigmoid activation
                                      function to inputs

    def tanh(self, x):
        return (torch.exp(x) - torch.exp(-x)) / (torch.exp(x) + torch.exp(-x)) #
                                             Computes the Tanh activation
                                             function

    def sigmoid(self, x):
        return 1 / (1 + torch.exp(-x)) # Computes the Sigmoid activation function

    def forward(self, x):
        x = self.tanh(self.hidden(x)) # Applies Tanh activation function to inputs
                                       and applies the hidden layer
        x = self.sigmoid(self.output(x)) # Applies Sigmoid activation function to
                                          inputs and applies the output
                                          layer

        return x # Returns the output

torch.manual_seed(190401008) # Setting the random seed of the torch module to my
                               student ID

net = Net()
output = net(X)
print(output)

```

Output:

tensor([[0.4929], [0.5002]])

4.3 (5 Puan) Kodlarınızın ve sonuçlarınızın olduğu jupyter notebook'un Github repository'sindeki linkini aşağıdaki url kısmının içine yapıştırın. İlk sayfada belirttiğim gün ve saate kadar halka açık (public) olmasın:

<https://github.com/OmerFarukAkyapak/yapaysiniraglari>

5 (Toplam 40 Puan) Multilayer Perceptron (MLP):

Bu bölümdeki sorularda benim vize ile beraber paylaştığım Prensesi İyileştir (Cure The Princess) Veri Seti parçaları kullanılacak. Hikaye şöyle (soruyu çözmek için hikaye kısmını okumak zorunda değilsiniz):

“Bir zamanlar, çok uzaklarda bir ülkede, ağır bir hastalığa yakalanmış bir prenses yaşamış. Ülkenin kralı ve kraliçesi onu iyileştirmek için ellerinden gelen her şeyi yapmışlar, ancak denedikleri hiçbir çare işe yaramamış.

Yerel bir grup köylü, herhangi bir hastalığı iyileştirmek için gücü olduğu söylenen bir dizi sihirli malzemeden bahsederek kral ve kraliçeye yaklaşmış. Ancak, köylüler kral ile kraliçeyi, bu malzemelerin etkilerinin patlayıcı olabileceği ve son zamanlarda yaşanan kuraklıklar nedeniyle bu malzemelerden sadece birkaçının herhangi bir zamanda bulunabileceği konusunda uyarılmışlar. Ayrıca, sadece deneyimli bir simyacı bu özelliklere sahip patlayıcı ve az bulunan malzemelerin belirli bir kombinasyonunun prensesi iyileştireceğini belirleyebilecekmiş.

Kral ve kraliçe kızlarını kurtarmak için umutsuzlar, bu yüzden ülkedeki en iyi simyacıyı bulmak için yola çıkmışlar. Dağları tepeleri aşmışlar ve nihayet "Yapay Sinir Ağları Uzmanı" olarak bilinen yeni

bir sihirli sanatın ustası olarak ün yapmış bir simyacı bulmuşlar.

Simyacı önce köylülerin iddialarını ve her bir malzemenin alınan miktarlarını, ayrıca iyileşmeye yol açıp açmadığını incelemiş. Simyacı biliyormuş ki bu prensesi iyileştirmek için tek bir şansı varmış ve bunu doğru yapmak zorundaymış. (Original source: <https://www.kaggle.com/datasets/unmoved/cure-the-princess>)

(Buradan itibaren ChatGPT ve Dr. Ulya Bayram'a ait hikayenin devamı)

Simyacı, büyü büyümlerinin farklı kombinasyonlarını analiz etmek ve denemek için günler harcamış. Sonunda birkaç denemenin ardından prensesi iyileştirecek çeşitli karışım kombinasyonları bulmuş ve bunları bir veri setinde toplamış. Daha sonra bu veri setini eğitim, validasyon ve test setleri olarak üç parçaya ayırmış ve bunun üzerinde bir yapay sinir ağı eğiterek kendi yöntemi ile prensesi iyileştirme ihtimalini hesaplamış ve ikna olunca kral ve kraliçeye haber vermiş. Heyecanlı ve umutlu olan kral ve kraliçe, simyacının prensese hazırladığı ilacı vermesine izin vermiş ve ilaç işe yaramış ve prenses hastalığından kurtulmuş.

Kral ve kraliçe, kızlarının hayatını kurtardığı için simyacıya krallıkta kalması ve çalışmalarına devam etmesi için büyük bir araştırma bütçesi ve çok sayıda GPU'su olan bir server vermiş. İyileşen prenses de kendisini iyileştiren yöntemleri öğrenmeye merak salıp, krallıktaki üniversitenin bilgisayar mühendisliği bölümüne girmiş ve mezun olur olmaz da simyacının yanında, onun araştırma grubunda çalışmaya başlamış. Uzun yıllar birlikte krallıktaki insanlara, hayvanlara ve doğaya faydalı olacak yazılımlar geliştirmişler, ve simyacı emekli olduğunda prenses hem araştırma grubunun hem de krallığın lideri olarak hayatına devam etmiş.

Prens, kendisini iyileştiren veri setini de, gelecekte onların izinden gidecek bilgisayar mühendisi prensler ve prensesler başkalarına faydalı olabilecek yapay sinir ağları oluşturmayı öğrensinler diye halka açmış ve sınavlarda kullanılmasını salık vermiş."

İki hidden layer'lı bir Multilayer Perceptron (MLP) oluşturun beşinci ve altıncı haftalarda yaptığımız gibi. Hazır aktivasyon fonksiyonlarını kullanmak serbest. İlk hidden layer'da 100, ikinci hidden layer'da 50 nöron olsun. Hidden layer'larda ReLU, output layer'da sigmoid aktivasyonu olsun.

Output layer'da kaç nöron olacağını veri setinden bakıp bulacaksınız. Elbette bu veriye uygun Cross Entropy loss yöntemini uygulayacaksınız. Optimizasyon için Stochastic Gradient Descent yeterli. Epoch sayınızı ve learning rate'i validasyon seti üzerinde denemeler yaparak (loss'lara overfit var mı diye bakarak) kendiniz belirleyeceksiniz. Batch size'ı 16 seçebilirsiniz.

5.1 (10 Puan) Bu MLP'nin pytorch ile yazılmış class'ının kodunu aşağı kod bloğuna yapıştırın:

```
# Define MLP class
class MLP(nn.Module):
    def __init__(self, input_size, hidden_size1, hidden_size2, output_size):
        super(MLP, self).__init__()
        self.hidden1 = nn.Linear(input_size, hidden_size1) # 1st hidden layer
        self.hidden2 = nn.Linear(hidden_size1, hidden_size2) # 2nd hidden layer
        self.output = nn.Linear(hidden_size2, output_size) # output layer

    def forward(self, x):
        x = torch.relu(self.hidden1(x)) # Apply ReLU activation function in the 1st hidden layer
        x = torch.relu(self.hidden2(x)) # Apply ReLU activation function in the 2nd hidden layer
        x = torch.sigmoid(self.output(x)) # Apply sigmoid activation function in the output layer
        return x
```



```

input_size = len(trainData.columns) - 1 # Number of input features, excluding the
                                         target column
hidden_size1 = 100 # Number of units in the 1st hidden layer
hidden_size2 = 50 # Number of units in the 2nd hidden layer
output_size = 2 # Number of output units (for binary classification)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu') # Use GPU if
                                         available, else use CPU

# Define the MLP model
model = MLP(input_size, hidden_size1, hidden_size2, output_size)
# Send the model to the device (GPU or CPU)
model = model.to(device)

```

5.2 (10 Puan) SEED=öğrenci numaranız set ettikten sonra altıncı haftada yazdığımız gibi training batch'lerinden eğitim loss'ları, validation batch'lerinden validasyon loss değerlerini hesaplayan kodu aşağıdaki kod bloğuna yapıştırın ve çıkan figürü de alta ekleyin.

```

# Seed ayarlama
torch.manual_seed(190401008)
# Batch size secilir
batch_size = 16
num_epochs = 50
train_loss_list = []
validation_loss_list = []

for epoch in range(num_epochs):
    # Train the model
    model.train() # train mode on
    train_loss = 0.0
    correct = 0
    total = 0
    for i, (inputs, labels) in enumerate(train_loader):
        # Move data to device
        inputs = inputs.to(device)
        labels = labels.to(device)

        # Forward pass
        outputs = model(inputs)
        loss = criterion(outputs, labels)

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        train_loss += loss.item() * inputs.size(0)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    # Print loss every few iterations
    if (i + 1) % 100 == 0:
        print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}")

    train_loss_list.append(train_loss / len(train_loader.dataset)) # add list for
                                                                    figure
    train_loss /= len(train_loader.dataset)
    train_acc = 100 * correct / total

# Evaluate the model on validation set every epoch
model.eval()
val_loss=0.0
with torch.no_grad():

```

```

correct = 0
total = 0
for inputs, labels in val_loader:
    # Move data to device
    inputs = inputs.to(device)
    labels = labels.to(device)

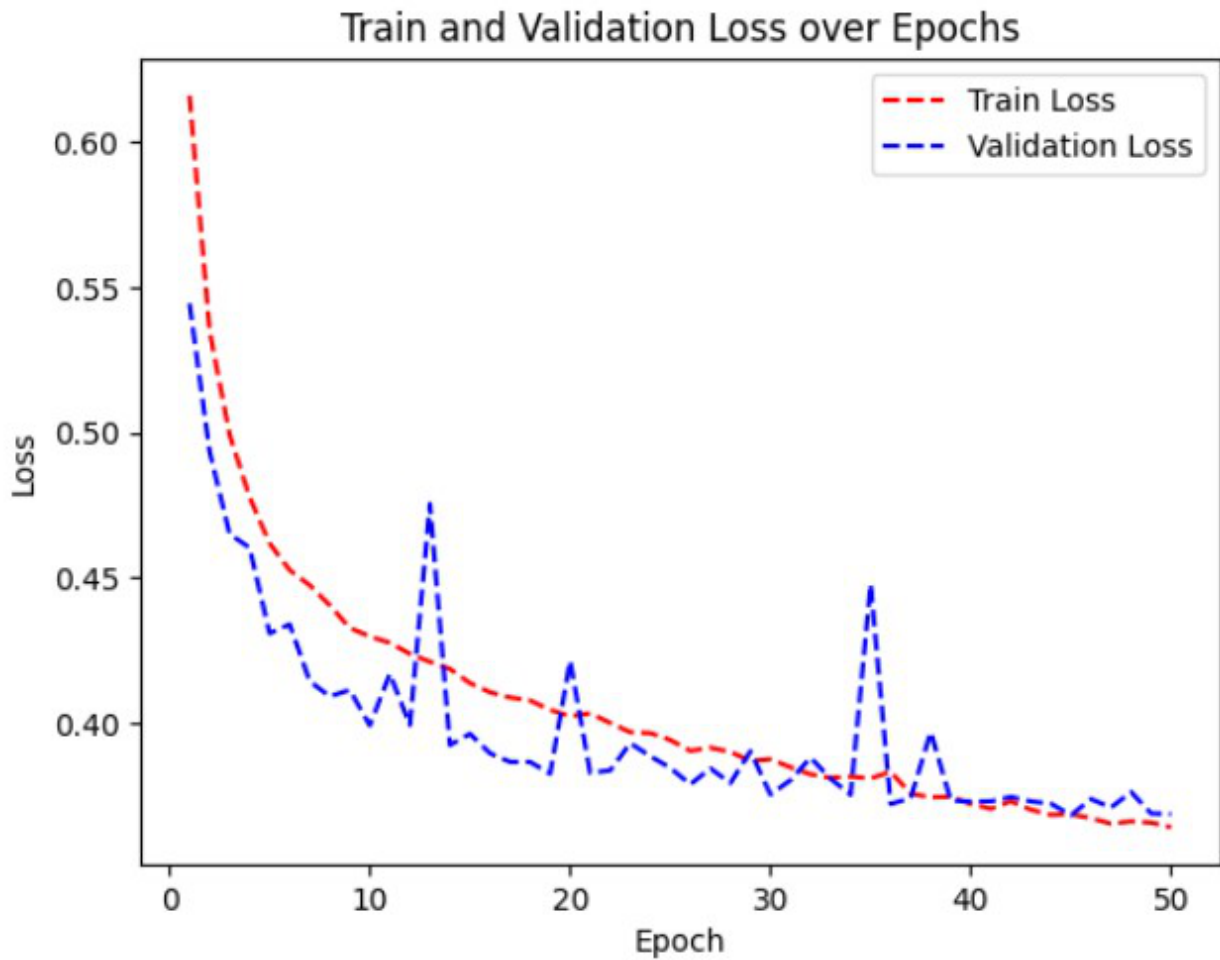
    # Forward pass and calculate accuracy
    outputs = model(inputs)
    loss = criterion(outputs, labels)
    # predicted = torch.round(outputs)
    # predicted = predicted.view(-1) Make output one-dimensional
    val_loss += loss.item() * inputs.size(0) # Collect Losses
    _, predicted = torch.max(outputs.data, 1)
    total += labels.size(0)
    correct += (predicted == labels).sum().item()

validation_loss_list.append(val_loss / len(val_loader.dataset))
val_loss /= len(val_loader.dataset) # Calculate the average by dividing the
                                   losses by the number of data

val_acc = correct / total
print(f"Epoch [{epoch+1}/{num_epochs}], Train Loss: {train_loss:.4f}, Train
      Acc: {train_acc:.2f}, Val Loss: {
      val_loss:.4f}, Val Acc: {val_acc:.
      4f}")

```

Output:



Şekil 1: 5.2 Train And Validation Losses

5.3 (10 Puan) SEED=öğrenci numaranız set ettikten sonra altıncı haftada ödev olarak verdiğim gibi earlystopping'deki en iyi modeli kullanarak, Prensesi İyileştir test setinden accuracy, F1, precision ve recall değerlerini hesaplayan kodu yazın ve sonucu da aşağı yapıştırın. %80'den fazla başarı bekliyorum test setinden. Daha düşükse başarı oranınız, nerede hata yaptığınızı bulmaya çalışın. %90'dan fazla başarı almak mümkün (ben dedim).

```
from sklearn.metrics import accuracy_score, precision_score, f1_score, recall_score

newModel = MLP(input_size, hidden_size1, hidden_size2, output_size).to(device)

# define best model path
model_path = '/content/checkpoint.pt'

# upload best model
newModel.load_state_dict(torch.load(model_path))

# Set the model in evaluation mode
newModel.eval()

# Loop over the test set and make predictions
with torch.no_grad():
    true_labels = []
    predicted_labels = []
    for inputs, labels in test_loader:
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = newModel(inputs)
        _, predicted = torch.max(outputs.data, 1)
        true_labels += labels.cpu().numpy().tolist()
        predicted_labels += predicted.cpu().numpy().tolist()

# Calculate performance metrics
accuracy = accuracy_score(true_labels, predicted_labels)
precision = precision_score(true_labels, predicted_labels)
recall = recall_score(true_labels, predicted_labels)
f1 = f1_score(true_labels, predicted_labels)

print(f"Test Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
```

Outputs:

Test Accuracy: 0.9145

Precision: 0.9086

Recall: 0.9227

F1 Score: 0.9156

5.4 (5 Puan) Tüm kodların CPU'da çalışması ne kadar sürüyor hesaplayın. Sonra to device yöntemini kullanarak modeli ve verileri GPU'ya atıp kodu bir de böyle çalıştırın ve ne kadar sürdüğünü hesaplayın. Süreleri aşağıdaki tabloya koyun. GPU için Google Colab ya da Kaggle'ı kullanabilirsiniz, iki ortam da her hafta saatlerce GPU hakkı veriyor.

Tablo 1: RUNTIME Süre (saniye)	
Ortam	
CPU	Code execution time: 0:00:05.147426
GPU	Code execution time: 0:00:10.617068

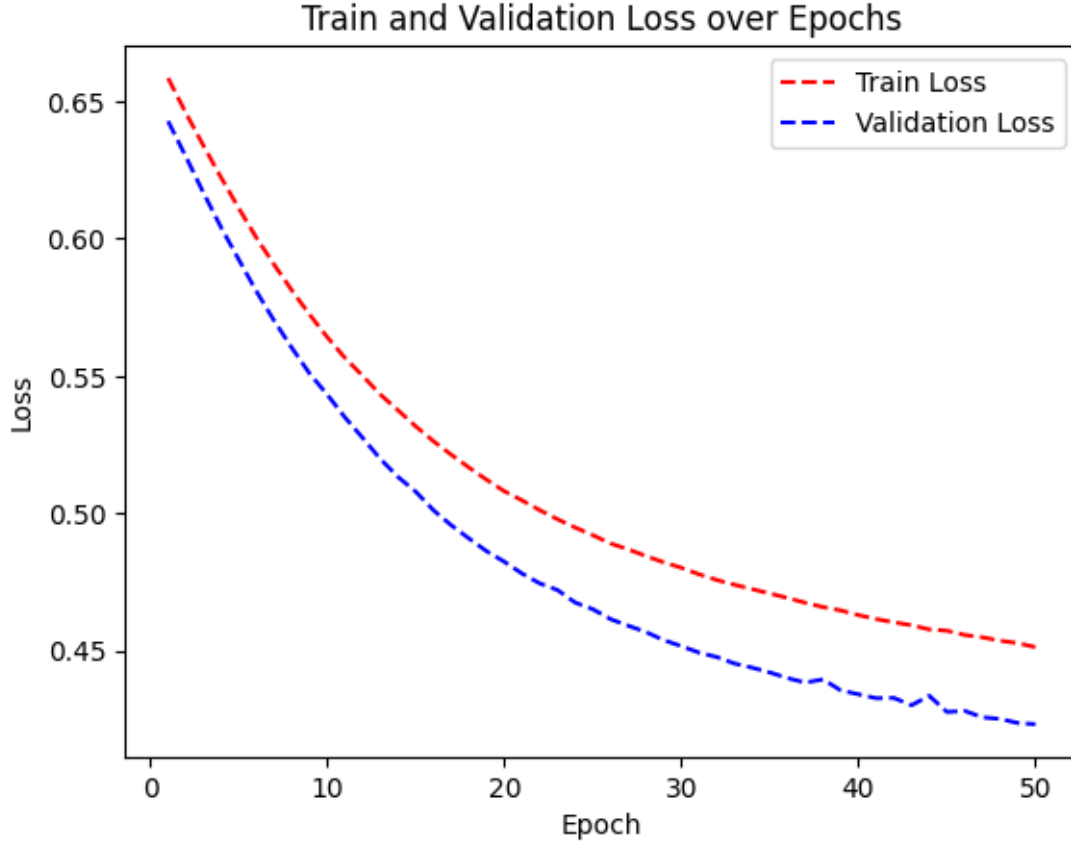
5.5 (3 Puan) Modelin eğitim setine overfit etmesi için elinizden geldiği kadar kodu gereken şekilde değiştirin, validasyon loss'unun açıkça yükselmeye başladığı, training ve validation loss'ları içeren figürü aşağı koyun ve overfit için yaptığınız değişiklikleri aşağı yazın. Overfit, tam bir çanak gibi olmalı ve yükselmeli. Ona göre parametrelerle oynayın.

Veri setini küçülttüm.

Early Stopping kullanmadım.

Model eğitim süresini arttırdım.

Learning rate değerlerini değiştirdim.



Şekil 2: 5.5 Train and Validation Losses

5.6 (2 Puan) Beşinci soruya ait tüm kodların ve cevapların olduğu jupyter notebook'un Github linkini aşağıdaki url'e koyun.

<https://github.com/OmerFarukAkyapak/yapaysiniraglari>

6 (Toplam 10 Puan)

Bir önceki sorudaki Prensesi İyileştir problemindeki yapay sinir ağına seçtiğiniz herhangi iki farklı regülerizasyon yöntemi ekleyin ve aşağıdaki soruları cevaplayın.

6.1 (2 puan) Kodlarda regülerizasyon eklediğiniz kısımları aşağı koyun:

```
##### L1 Regularization #####
l1_lambda = 0.001
l1_reg = torch.tensor(0.)
for param in model.parameters():
    l1_reg += torch.norm(param, p=1)
loss += l1_lambda * l1_reg
```

```
#EarlyStopping
val_score=val_loss

if best_val_loss is None:
    patience_counter=0
    best_val_loss=val_score          # Keep patience length in memory
    torch.save(model.state_dict(),"checkpoint.pt")
elif best_val_loss<val_score:      # patience counter
    patience_counter+=1
    print("Earlystopping Patience Counter",patience_counter)
    if patience_counter==patience:
        break
else:
    best_val_loss=val_score
    torch.save(model.state_dict(),"checkpoint.pt")          # keeps the best
                                                             model
    patience_counter=0          # reset patience after we get the best model
```

6.2 (2 puan) Test setinden yeni accuracy, F1, precision ve recall değerlerini hesaplayıp aşağı koyun:

Accuracy: 0.9093 ————— Accuracy: 0.9378
 Precision: 0.9569 ————— Precision: 0.9775
 Recall: 0.8582 ————— Recall: 0.8969
 F1 Score: 0.9049 ————— F1 Score: 0.9355

6.3 (5 puan) Regülerizasyon yöntemi seçimlerinizin sebeplerini ve sonuçlara etkisini yorumlayın:

L1 regularizasyonu ile modelin ağırlıklarını düzenlemek amacıyla kullandım. Gereksiz özelliklerin seçilmesini veya ağırlıkların küçültülmesini sağladım. Yani modelde sadece önemli özelliklere odaklanıldım.

Buna ek olarak kodum da EarlyStopping de kullandım tam anlamıyla bir regularizasyon yöntemi olmasa da modelin aşırı uyum (overfitting) riskini azaltmak için kullanıldım. Bu yöntemler sonucunda değerlerimin yeterince yükseldiğini gözlemledim.

L1 regularizasyonu L2 ye kıyasla daha verimli çalıştı. Githubda L2 regularizasyonumda bulunmaktadır.

6.4 (1 puan) Sonucun github linkini aşağıya koyun:

<https://github.com/OmerFarukAkyapak/yapaysiniraglari>